

四阶段完整分析策略：

第一阶段：仓库演化与社区画像分析（数据层面）

- **分析工具：**PyDriller, GitPython, Pandas, Matplotlib。
 - **执行策略：**
 - **贡献者分布：**分析 Commit 记录，画出“贡献者活跃度曲线”。识别谁是核心开发者，分析社区代码是由少数人控制还是高度去中心化。
 - **模块稳定性分析：**统计不同文件夹（如 backend, frontend, ui）的修改频率。频率最高的地方通常是 Bug 密集的“热点区域”。
 - **Issue 生命周期：**爬取 GitHub Issue 数据，计算 Bug 从提出到关闭的平均时长，分析 MaxKB 维护团队的响应速度。
-

第二阶段：基于 LibCST 的静态代码分析（技术难度层面）

这是课程明确提到的加分项，重点在于**不运行代码就能发现潜在问题**。

- **分析工具：**libcst, flake8, bandit（安全扫描）。
 - **执行策略：**
 - **代码异味检测：**编写一个 libcst.Visitor 脚本，专门扫描 MaxKB 中不规范的异步写法（例如在 Django 视图中错误地使用了同步阻塞调用）。
 - **自动重构实验：**尝试编写一个 libcst.Transformer，将项目中的旧式字符串格式化（如 % 或 .format()）批量转换为 Python 3.6+ 的 f-string。
 - **复杂度扫描：**利用 ast 库计算核心业务逻辑（如 RAG 检索逻辑）的圈复杂度（Cyclomatic Complexity），找出最难维护的函数。
-

第三阶段：动态分析与模糊测试 Fuzzing（寻找 Bug 层面）

这是最容易通过“发现真实 Bug 并提交”获得额外加分的环节。

- **分析工具：**Atheris（针对 Python 的 Fuzzer）或简单的 Python-Hypothesis。
 - **执行策略：**
 - **文件解析测试：**MaxKB 支持上传 PDF、Markdown 和 Excel。这些文件解析库往往存在边界情况漏洞。构造畸形文件上传，观察后台是否抛出 500 错误或内存溢出。
 - **API 随机压力测试：**针对 knowledge_base 的创建、检索 API，使用随机参数进行探测，寻找未授权访问或逻辑越权。
 - **加分动作：**若发现崩溃（Crash），立即在 GitHub 提交 Issue。**在文档中附上 Issue 链接是最高级的证明。**
-

第四阶段：基于 Z3 的逻辑形式化验证（学术深度层面）

针对 MaxKB 的核心逻辑进行数学建模。

- **分析工具：**z3-solver。
- **执行策略：**
- **权限模型验证：**MaxKB 拥有“租户-角色-资源”的权限体系。

- **实现方法：**将权限规则转化为 Z3 的约束表达式 (Constraints)。例如：`Assert(UserRole == 'Guest' && Resource == 'PrivateDB' => Access == 'Denied')`。
 - **寻找矛盾：**利用 Z3 求解器搜索是否存在一种状态，使得低权限用户能够访问高权限资源。
-

团队分工与 GitHub 管理建议

为了体现“团队协作”，请务必遵守以下操作：

1. **Fork 仓库：**组长先 Fork [1Panel-dev/MaxKB](#) 到自己的账户，然后邀请组员作为 Collaborator。
 2. **分支策略：**
 - `analysis/evolution`: 存放数据分析脚本和图表。
 - `analysis/static`: 存放基于 `libcst` 的代码扫描工具。
 - `analysis/fuzzing`: 存放测试脚本。
 3. **提交规范：**
 - 每个人必须有独立的 Commit 记录。
 - Commit Message 使用规范格式，例如：`feat: add libcst visitor for async detection`。
-

最后的提交文档大纲（毕设标准）

- **第 1 章 绪论：**分析 MaxKB 项目的背景、开源协议 (GPL-3.0) 及其在 LLM 生态中的地位。
- **第 2 章 仓库演化分析：**展示第一阶段的图表，分析开发周期和社区活跃度。
- **第 3 章 静态代码质量分析：**展示 LibCST 扫描结果，列出发现的代码缺陷。
- **第 4 章 安全性与鲁棒性测试：**记录模糊测试过程，展示发现的 Bug 或潜在风险。
- **第 5 章 逻辑建模与验证：**展示 Z3 求解器的代码和逻辑推演结果。
- **第 6 章 总结与贡献：**总结组员贡献，附上 GitHub PR 或 Issue 链接。